# Bolt Beranek and Newman Inc.

bbn

2

AD-A151 312

Report No. 5883

# Combined Quarterly Technical Report No. 35

Pluribus Satellite IMP Development
Mobile Access Terminal Network

November 1984

Prepared for:
Defense Advanced Research Projects Agency







85 03 05 010

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER 34 | 2. GOVT ACCESSION NO. AD-A151 312 | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle) Combined Quarterly Technical Report No. 35 | | 5. TYPE OF REPORT & PERIOD COVERED Quarterly Technical 8/1/84 - 10/31/84 |
| | | 6. PERFORMING ORG. REPORT NUMBER 5883 |
| 7. AUTHOR(s) | | 8. CONTRACT OR GRANT NUMBER(s) MDA903-80-C-0353 N00039-81-C-0408 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 10 Moulton Street Cambridge, MA 02238 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209 | | 12. REPORT DATE November 1984 |
| | | 13. NUMBER OF PAGES 33 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) DSSW Room ID The Pentagon Washington, DC 20310 | NAVELEX Washington, DC 20360 | 15. SECURITY CLASS. (of this report) UNCLASSIFIED |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE/DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer networks, packets, packet broadcast, satellite communication, gateways, Pluribus Satellite IMP, shipboard communications, ARPANET, Internet

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This Quarterly Technical Report describes work on the development of Pluribus Satellite IMPs, and on shipboard satellite communications.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Report No. 5883

COMBINED QUARTERLY TECHNICAL REPORT NO. 35

PLURIBUS SATELLITE IMP DEVELOPMENT
MOBILE ACCESS TERMINAL NETWORK

November 1984

| Accession For | |
|---|---|
| NTIS GRA&I | X |
| DTIC TAB | ☐ |
| Unannounced | ☐ |
| Justification | |
| By | |
| Distribution/ | |
| Availability Codes | |
| Dist | Avail and/or Special |
| A-1 | |

This research was supported by the Defense Advanced Research Projects Agency under the following contracts:

MDA903-80-C-0353, ARPA Order No. 3214
N00039-84-C-0408

Submitted to:

Director
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209

Attention: Program Management

Table of Contents

# 1 INTRODUCTION

This Quarterly Technical Report is the current edition in a series of reports which describe the work being performed at BBN in fulfillment of several ARPA work statements. This QTR covers work on several ARPA-sponsored projects including (1) development of the Pluribus Satellite IMP, and (2) development of the Mobile Access Terminal Network. This work is described in this single Quarterly Technical Report with the permission of the Defense Advanced Research Projects Agency. The work on the Mobile Access Terminal Network under contract 0408 has been completed. Some of this work is a continuation of efforts previously reported on under contracts DAHC15-69-C-0179, F08606-73-C-0027, F08606-75-C-0032, MDA903-76-C-0214, MDA903-76-C-0252, N00039-79-C-0386, and N00039-78-C-0405, and N0039-81-C-0408.

## 2 PLURIBUS SATELLITE IMP DEVELOPMENT

During the quarter, BBN concentrated its efforts on a number of areas. On August 9, BBN hosted a meeting of the Wideband Network community. At that meeting, it was decided to operate the network in a "quasi-operational" mode on Thursdays and Fridays of each week. Following that meeting, BBN began spending an increased amount of its effort on Wideband Network operations. During the non-operational periods, much progress was made in Wideband Network systems integration. BBN identified and corrected a PSAT software bug which had been preventing the network from operating with more than four channel streams. Western Union completed the installation of three additional earth stations at M/A-COM Linkabit, CMU, and BBN during the quarter.

The BSAT software running in the Lincoln Voice Funnel hardware was successfully operated on satellite channel using the PSAT Translator. A small Butterfly system was shipped to M/A-COM Linkabit during the quarter and progress was made in the integration of the BSAT with the ESI-B.

In addition to the testing with the PSAT translator and ESI-B, progress was made in other aspects of the BSAT software. During the quarter, BSAT code which maintains datagram reservation synchronization was implemented. This code is necessary to allow

multisite BSAT testing to begin.

## 2.1 Wideband Network Operations

The network was down during the period of July 27 - August 6 due to the satellite changeover from WESTAR III to WESTAR IV. During August, the noise environment found on WESTAR IV was significantly better than was encountered on WESTAR III, leading to improved Wideband Network performance. The entire network was down again during the period August 20-21 while Western Union conducted acceptance tests of the new earth stations at M/A-COM, Linkabit, and CMU.

BBN hosted a meeting of Wideband Network community on August 9. The Wideband network task force reported on their most recent progress. At this meeting, it was determined that the system integration effort had proceeded to the point where, although there were still a few outstanding problems which needed to be resolved, the network was stable enough to be operated on a quasi-operational basis for two days each week. During that time, BBN would make every effort to keep the network up, running, and avilable to the users. In particular, this included the ISI Switched Telephone Network Interfaces (STNIs), the Lincoln Packet Voice Terminals (PVTs), and Miniconcentrator Gateways at the four major DARPA sites; Lincoln Laboratory, ISI,

SRI, and DCEC.

BBN spent an increased proportion of its efforts on network operations during August and September following the decision to maintain an operational network and make it available to users on Thursdays and Fridays of each week. The primary source of user traffic was voice calls between switched telephone network interfaces (STNIs) at the four main DARPA sites (ISI, Lincoln, DCEC, and SRI). BBN expanded its network monitoring to include periodic manual testing of the STNI equipment at each site during the operational periods.

A significant problem hindering network operations during the quarter was a PSAT software bug which limited the number of channel streams that could be supported in the network to four. Creation of a fifth channel stream would cause most of the sites in the network to crash. On the non-operational days, BBN mounted a significant effort aimed at identifying the source of this problem and correcting it. The requirements to maintain operational service two days/week required that some of the PSATs be patched, so that they would not create streams. Maintaining the patches across SAT software reloads proved to be a formidable task and there were several brief network outages during the quarter due to the "5 stream bug".

There were several hardware problems during August and September.

The Ft. Huachuca site was off the air during most of August due to problems with their ESI. Lincoln experienced a brief outage on August 14 due to an ESI problem, and the ESI at ISI experienced hardware problems during the period of August 13-15. The Lincoln site was powered down during the period August 24-29 due to lab renovations. During this period, the PSAT, Voice Funnel, and ESI were relocated to an area of the lab which provided better cooling and equipment accessibility.

The HPA at DCEC failed during the week of September 10th and was repaired by Western Union at the beginning of the following week. During a period of heavy laboratory construction in the computer room at Lincoln, on September 13th, the Miniconcentrator gateway failed intermittently. Reseating the boards in the PDP-11 seemed to clear up the problem. The PVT at SRI failed on September 20th and the PVT was repaired at the beginning of the following week. The ESI at Lincoln failed on September 20th and it was also repaired at the beginning of the following week. The Lincoln PSAT failed on September 27th. It was repaired and operational by October 1st. On September 28th the SRI ESI and the DCEC Minconcentrator gateway failed. Both of these were repaired during the following week.

In light of the problems limiting operational service, a six-week hiatus in operations was requested of DARPA to allow some of these problems to be worked on in a more concentrated manor. The

six-week period began with the week of October 1 and will end with the week of November 14.

## 2.2 Wideband Network Systems Integration

Early in the six-week hiatus period, the cause of the 5 stream bug was discovered. A dynamic data structure used for stream scheduling was not initialized to a sufficiently large size to support more than four channel streams. However, the expansion of this structure to a size sufficient to handle the large number of anticipated streams overflowed the common variables page of PSAT memory where this structure resides. As a result, a large amount of memory and code reorganization was required to fix this problem. A new PSAT software version containing this reorganization was released on 24 October. This release also contained modifications to expand the control subframe to nine slots to allow the addition of the Linkabit, CMU, and BBN sites. Also included in this release was a reduction in interburst padding from 1024 symbols to 768 symbols. Despite the increase in the size of the control subframe required for the addition of Linkabit, CMU and BBN, the concurrent reduction in interburst padding caused only a 9 percent loss in the length of the portion of the PODA frame available for streams and datagrams.

Another problem which hindered operations concerned the sensing

of Touch-Tone digits at the DCEC and SRI STNIs. Numbers dialed into the STNI would frequently be lost, thus preventing call completion even though systems beyond the STNI were functional. ISI investigated this problem and identified the cause. A modification has been installed in all the STNI cards to increase the signal level at the input to the Touch-Tone decoder. This should improve the reliability of tone reception. In particular, the reception at DCEC shows significant improvement as tested by long-distance calls. At SRI, one local phone whose tones weren't received before now works, but when the STNI is called from ISI via AT&T, the digits 1,2,3 are not heard even though the other digits are heard. This indicates a low-frequency roll-off somewhere in that path. Calls via MCI work fine.

The two STNIs at ISI are now directly accessible both by outside numbers and ISI extensions. A new version of the yellow dialing card will be issued to reflect this change and other improvements.

Another problem which was observed involves PSAT/ESI communication. At apparently random intervals, the PSAT will declare that the clock signal, provided by the ESI and used for all global time computations, has stopped. This clock is central to all time-keeping and scheduling functions and its cessation forces immediate restart of the PSAT. This problem has a major impact on network stability and, with the fix of the Five Stream

Bug, is seen as the most significant block to stable multisite (i.e., more than 5 sites) operation. This condition appears to occur more frequently under deteriorated channel conditions (or perhaps with sites whose Earth Terminal equipment is out of calibration) and has been observed to occur simulataneously at two or more sites.

The problem of global time not advancing was observed frequently at BBN when its Earth Terminal came on-line. Investigations revealed that the LTCLOCK signal in question was heavily distorted. The source of this distortion was found to be incorrect terminations at the SMI end of the cable connecting the ESI to the PSAT. The frequency of global time clock stoppage at BBN was reduced to nearly zero when this was fixed. However, subsequent checks of other sites plagued by this problem revealed no incorrect terminations.

Linkabit has been looking in detail at the LTCLOCK signal with a logic analyzer. BBN has inserted several patches into the PSAT code in an effort to get a better feeling for what is causing this problem. At this point, the problem is occurring infrequently and little progress has been made in tracking it down.

## 2.3 BSAT Software Development

During the quarter two significant milestones were achieved in the development of the BSAT. BBN completed and debugged the PSAT Translator program which, when run in the PSAT hardware, allows a BSAT running in Voice Funnel hardware to be connected to an ESI-A and to operate over the satellite channel.

BBN devoted some of its efforts on non-operational days during September to PSAT Translator debugging and BSAT software development. On October 3rd, a BSAT program running in the Voice Funnel machine at Lincoln was able to successfully talk through the PSAT Translator to the ESI-A and over the channel for the first time. The BSAT reset the ESI, commanded it to acquire the gross frequency offset (GFO), and proceeded to send round-trip-time ranging packets and leader packets. The test configuration was as follows:

```
|--------|         |----------|         |-------|         |--------|
|  BSAT  | ----->  |   PSAT   | ----->  |       | ----->  | Earth  |
| (Voice |         |Translator|         | ESI-A |         |        |
| Funnel)| <-----  |  (PSAT)  | <-----  |       | <-----  |Terminal|
|--------|         |----------|         |-------|         |--------|
```

At the end of August, a 4 processor Butterfly system was shipped to Linkabit in San Diego for the integration of the BSAT with ESI-B under development at Linkabit. In order to fit the BSAT software into a 4 processor machine, the host interface code was removed and the memory of one of the processor nodes was

increased from 256K bytes to 1 Mbyte by replacing the 64K dynamic RAM chips with 256K dynamic RAMs. This 4 processor prototype BSAT included one Butterfly synchronous I/O board. These I/O boards have 4 synchronous HDLC ports which can operate up to a maximum data rate of 2Mb/s and provide a 16 bit CRC checksum over the link between the BSAT and ESI-B. BBN expects shortly to receive a contract to develop a new satellite modem interface for the BSAT (BSMI). The BSMI will be capable of operating a single full duplex HDLC linkup to 4 Mb/s and will provide a 32 bit CRC for error detection over the satellite channel on top of the 16 bit CRC on the BSAT to ESI-B link.

Reservation synchronization has been implemented in the BSAT and initial debugging of the code has been performed. Further testing will be done with multiple BSATs connected to the BSAT Satellite Simulator. A detailed description of the reservation synchronization implementation, emphasizing the operation of process Reservation_Sync and related design issues, is included in a later section of this report.

A significant number of additions to the BSAT Channel Module, other than the implementation of process Reservation_Sync, were required in order to support the reservation synchronization function. Most of these additions were in process Scheduler. New scheduler features include:

- o Scheduling (but not yet transmitting) fragmented datagram bursts
- o Reporting datagram scheduling information to process Reservation_Sync
- o Receiving scheduling error information from Reservation_Sync and using it to adjust current scheduling
- o Dynamically adapting all PODA subframe scheduling and burst transmissions to the BSAT's current stream and reservation synchronization status on the channel
- o Implementing the CPODA protocol, including control subframe expansion.

Other Channel Module additions included code which was added to the downlink processes to pass information on received datagram transmissions to process Reservation_Sync, and a mechanism allowing the scheduler and stream aggregator processes to insert any required ranging or reservation information into outgoing bursts without waiting on locks.

Program trap signalling was installed throughout the BSAT, and trap information collecting code was installed in the Monitor Host. Some of this code was taken from code written for the Voice Funnel. Traps provide a way for a process to signal the occurrence of unusual, important, or interesting events. The trap mechanism installed allows a variable number of arguments to

be passed along with the identifying trap number. The trap number is used to indicate the relative importance of the trap. A trap reporting threshold mechanism was also installed.

A Demonstration Mode was added to the terminal command process. This displays a picture of the major modules of the BSAT and shows the flow of host messages and channel bursts. The display is updated every few seconds. Throughputs are shown both in messages or bursts per second and cumulative totals. The number of messages on the queues interconnecting the modules and the length of some important queues in the channel module are also shown. Using this display, one can see much of the system that is usually invisible. This includes the flow of messages to and from a host such as the Voice Funnel, aggregation of host messages into channel bursts, and the distribution of messages from the channel for this site versus those not for this site.

Several Chrysalis bugs, which had appeared as bugs in the BSAT and the Voice Funnel, were found and fixed. These bugs caused buffers to be lost when a queue of free buffers was nearly empty. This problem had prevented the BSAT from running at high traffic loads (when there are little or no free buffers in some parts of the system). Tests made after the Chrysalis bugs were fixed showed that buffers no longer mysteriously disappeared at any traffic load.

## 2.4 Reservation Synchronization Implementation in the BSAT

The implementation of the PODA datagram message scheduling function in the BSAT via a distributed control mechanism causes the component of datagram message delivery delay, attributable to satellite channel propagation time, to be equal to two 1/4-second satellite hops. Although such performance is superior to the corresponding three satellite hop delay resulting from centralized control, one of its costs is the need to allocate BSAT processing bandwidth to the detection and subsequent correction of datagram scheduling errors caused by datagram reservations that are heard on the satellite channel by some sites and are missed, due to channel noise, by others. This detection and correction of datagram scheduling inconsistencies is referred to as the maintenance of reservation (or datagram) synchronization. Another situation occurs when a site first joins ongoing distributed control operations on a satellite channel. Such a site does not have on its scheduling queue any as yet unserviced datagram reservations that were previously enqueued on the scheduling queues of the other channel sites. The newly operational site's BSAT must perform processing similar to that mentioned above in order to bring its scheduling queue into agreement with the other sites' queues, at with point it can initiate the transmission of datagrams to hosts at the other sites. This is known as the acquisition of reservation

synchronization.

The method used by the BSAT for the acquisition and maintenance of reservation synchronization is essentially the same as that used by the PSAT, although implementation details differ between the two systems. Although the following description of the BSAT implementation may explain some aspects of the technique, it is not intended to be tutorial and it assumes that the reader has already perused other documentation describing reservation synchronization in the Wideband Network. Among the relevant documents are: "Control Issues in a PODA Voice/Data Satellite Network," E. Killian and R. Binder, International Conference on Communications, June 1980, which describes the modifications made to the SATNET reservation scheduling/synchronization technique in order to efficiently support the Wideband channel; "PSAT Technical Report," Falk, et al., BBN Report No. 4469, May 1981; and "Datagram Scheduling Synchronization," Earl Killian, W-Note No. 20, August 1979, which, although now incorrect in some minor details, is informative for its pseudo-code description of the basic datagram scheduling/synchronization algorithms which remain unchanged. In the following description, reservation and stream synchronization will be abbreviated as res-sync and stream-sync.

The majority of the BSAT processing related to the reservation synchronization function for a satellite channel is performed by process "Reservation_Sync" and most of the remainder of such

processing is performed by process "Scheduler." The breakdown of the res-sync functions between these two processes has been designed and implemented with the goal of eliminating the need for any inter-process locking mechanisms that would require either process to wait on a lock, while at the same time keeping as much of the res-sync task out of the scheduler as possible. This design goal is significant in light of the fact that the pipelined nature of datagram scheduling permits the existence of only one scheduler/res-sync process pair for each independent satellite channel module present in a BSAT. It is particularly critical that a channel's single scheduler process not be delayed or diverted from its primary task i.e., scheduling time in each 21 msec PODA frame for datagram and other types of bursts and queueing any such bursts originating from the local site to uplink processing soon enough so that the bursts reach the local ESI in time for transmission. Having the scheduler wait on locks and/or perform res-sync processing that could be performed elsewhere is contrary to this goal.

The process Reservation_Sync communicates with: [1] all other channel protocol module (CPM) processes via a set of flags kept in a "status" variable in the CPM's common memory segment, [2] the channel's scheduler process via a "scheduled datagrams to sync" dual queue, [3] the channel's process(es) which reconstruct messages from received rearranged bursts via a "received

datagrams to sync" dual queue, and [4] the (same) scheduler process via a "scheduling error" structure in the CPM common memory segment. The details of these communication mechanisms and an overview of the operation of the res-sync process are presented in the following subsections.

"Datagram burst," as used in this description, refers to BOTH complete datagram bursts and the first fragment of datagram bursts which must be transmitted in multiple datagram subframes. No blocks of information are sent to the res-sync process corresponding to the transmitted or received burst fragments required to continue and/or terminate a datagram transmission. This is the case because it is only the STARTING channel time and extent of a datagram burst (fragmented or not) allocated in the schedulable part of the datagram subframe which is globally known and must therefore remain synchronized among all of the sites on the channel. Sufficient time is provided on the channel by all of the the sites to transmit all components of a fragmented burst; however, the exact transmission times of any trailing fragments are only determined by the transmitting site.

### 2.4.1 Inputs to Reservation_Sync

#### 2.4.1.1 Use of the CPM Status Variable

The CPM status variable contains a number of single-bit flags which indicate the state (in or out) of each type of CPM synchronization (frame, group, stream, and reservation), whether or not the site considers itself to be leader on the channel, and the site's leadership eligibility. Different CPM processes read and/or write different flags, using Chrysalis-provided atomic operations to consistently update the status variable. The res-sync process is concerned only with the stream-sync and the res-sync flags. It reads the stream-sync flag, which it assumes to be controlled by other CPM processes, every time it makes a processing pass. Since the acquisition and maintenance of res-sync presupposes a site's knowledge of the location of the start of each datagram subframe, which is, in turn, dependent on being in stream-sync, the res-sync process performs no significant action when it finds that the site is out of stream-sync. If the site is found to be in stream-sync, however, the res-sync process will perform its primary function of detecting any datagram scheduling errors made by the scheduler in the past (about one round-trip time ago) and passing information regarding such errors on to the scheduler for its use in the adjustment of current scheduling.

It is assumed that all one-to-zero-to-one transitions of the stream-sync flag in the CPM status variable, indicating a loss and subsequent re-acquisition of stream-sync, are of sufficient duration to be detected by those CPM processes (e.g., the scheduler process and the res-sync process) that must reset their status upon stream-sync state changes. Tn most cases, the time needed for a site to request the stream database and receive it from another site over the satellite channel will be of more than sufficient duration to fulfill this requirement However, if a site is able to quickly reacquire stream-sync by receiving a steam database transmission already requested by another out of stream-sync site, the CPM processes controlling the stream-sync flag must still guarantee the minimum out of stream-svnc interval. It is also assumed that any process altering the stream-sync flag will always simultaneously clear the res-sync flag.

In addition to providing the scheduler with scheduling error information, the res-sync process controls all transitions into and out of the res-sync state (via the res-sync status flag) as long as the site is in stream-sync on the channel. The res-sync status flag's setting allows or inhibits local BSAT transmissions of datagram and setup bursts on the channel. The one exception is the case where a site comes up on a channel on which no other sites are active. In that case the scheduler process declares

the site to be leader and to be in every kind of channel synchronization, including res-sync. The res-sync process is designed to recognize this situation and handle it properly.

The previous paragraph notes that the res-sync process controls all transitions, BOTH into and out of the res-sync state, while the site is in stream-sync on the channel. The latter type of transition may not seem necessary given the BSAT's technique for acquiring and maintaining res-sync, which is designed to assure the continuation of the in res-sync state as long as the boundaries of every datagram subframe are known (i.e., stream-sync is maintained) and current scheduling corrections are applied as past scheduling errors are detected. Certain unusual occurrences on a noisy satellite channel. however, can cause the res-sync technique to fail. An example of such an occurrence would be the case where one site, "site 1, receives a datagram reservation that none of sites 2-N manage to receive, where N>>1 and the channel is always busy (there are always datagrams to schedule). In that case site 1 would always be scheduling datagrams later than sites 2-N and its datagram transmissions would interfere with those from sites 2-N. Site 1 would not make any scheduling corrections, however, since it would believe that all of sites 2-N require such adjustment. Sites 2-N would never correct to site 1, since they would not receive any of site 1's colliding transmissions as a reference. Such a lockup could

continue as long as the channel did not empty out. As a precaution against this and other rare and pathological datagram scheduling conditions, the res-sync process implements a feature whereby it will force a loss and subsequent re-acquisition of res-sync if a site which believes itself to be in res-sync on a non-empty channel does not perform a single verified correct scheduling in a stated time interval (on the order of seconds). The implementation of this feature has a negligible processing cost.

#### 2.4.1.2 Scheduled Datagrams to Sync Queue

The "scheduled datagrams to sync" queue is the means by which the res-sync process acquires information about the datagram scheduling decisions that have been made by the scheduler process. The entities on this queue are the buffer IDs of the head buffers of chains of datagram reservation blocks linked via their "buf next" fields. Every chain contains a reservation block corresponding to each and every datagram burst scheduled by the scheduler during a particular PODA frame. The scheduler only schedules datagrams and enqueues the block chains to the res-sync process when the site is in stream-sync on the channel. Enqueueing chains of reservation blocks, rather than individual blocks, both reduces the required size of the interprocess dual queue and limits the number of enqueue operations on the queue to

one per frame. The delay this method may cause in the receipt of the reservation blocks by the res-sync process is insignificant because the information in the blocks is not generally useful to the process until approximately one round-trip time after the corresponding reservation has been scheduled.

The scheduling information in each datagram reservation block that is required by the res-sync process is as follows: the sending site and burst ID of the scheduled datagram burst, which are used to uniquely identify the burst when (and if) it is received from the satellite downlink; and the starting channel (global) transmission time that was scheduled for the burst, which is used with the observed starting transmission time of a received and matched burst in verifying correct scheduling or calculating scheduling errors In addition to the above per-block fields, the res-sync process requires the scheduler to insert the value of the cumulative non-datagram-subframe channel time for a frame in a field in the first block of the corresponding chain. This value is used in the calculation of datagram scheduling errors when the scheduled and observed datagram bursts are in different frames. If the CPM is in stream-sync and the scheduler doesn't schedule any datagram reservations in a given frame, it queues a single "dummy" reservation block to the res-sync process containing the non-datagram-subframe time for the frame as well as the channel time

of the start of the frame.

When the res-sync process acquires a chain of datagram reservation blocks while in the stream-sync state, it first records the channel time of the start of the corresponding frame and saves the cumulative non-datagram-time value for the frame in an array of such values maintained for (currently) the last 32 frames. The required scheduling information from each datagram reservation block on the chain is then block transferred into a separate element on a local queue of scheduling information maintained within the res-sync process's initialized data segment. The reservation blocks are freed as they are copied. The local queue's elements are linked together in the same order as the corresponding datagrams were scheduled. The elements kept on the queue are up to 32 frames old. The local queue is used to store scheduling information for a number of efficiency reasons. Since a single block pool is used by many CPM processes for a number of different purposes, it is most efficient not to tie up 1/4 second or more of datagram reservation blocks in the res-sync process while each block waits for its corresponding datagram burst to be received. Since the local queue elements are not buffers with the latters' consequent system overhead, they can store a given amount of scheduling history using much less memory than blocks can. Finally, searching the local queue for the element corresponding to a received datagram burst does not

involve switch accesses; freeing matched or otherwise unneeded elements from the queue does not involve dual queue operations.

### 2 4.1.3 Received Datagrams to Sync Queue

The "received datagrams to sync" queue is the means by which the res-sync process acquires information about the datagram bursts that have been received from the satellite downlink The entities on this queue are the buffer IDs of datagram burst blocks; they are placed on the queue by the deconvolver processes that are part of the given channel module. Each datagram burst block contains the sending site and burst ID extracted by a deconvolver from a single received datagram burst, and the observed channel time of the burst. The latter value is calculated by a deconvolver from the ESI time of the burst's arrival. the current round-trip time value for the channel, and the current ESI time to channel time offset for the channel.

In addition, each datagram burst block contains a sequence number. The single "CPM_Rcve" process for the channel keeps the master copy of the sequence number, which it increments by one and inserts into the "buf flags" field of the burst descriptor of every error-free datagram burst that it receives from the channel. The deconvolvers simply copy the sequence numbers into the datagram burst blocks before queueing them to the res-sync

process. Since, in the case of multiple deconvolvers in a single CPM, datagram burst blocks can arrive out of order at the CPM's res-sync process, the sequence number allows the latter process to establish the proper ordering of the blocks. It should also be noted that a deconvolver will simply discard a datagram burst block without comment (via traps. throws. etc.), if it finds that the dual queue to the res-sync process is full. This is because the res-sync technique used does not require the processing of every datagram burst heard on the channel; allocating the necessary BSAT processing bandwidth, queue sizes, block pool sizes, etc. to process every burst under conditions of maximum channel loading may in fact be undesirable.

### 2.4.2 Reservation_Sync Operation and Outputs

With the specification of the inputs to the res-sync process completed, an overview of the process's operation and outputs can be given.

#### 2.4.2.1 Main Processing Loop

The res-sync process is runnable and makes a single processing pass every time it dequeues a new datagram burst block from the "received datagrams to sync" dual queue. When a processing pass

is completed and there are no new blocks waiting on the queue, a timer is set to guarantee that the process will again run a processing pass after (currently) 21 milliseconds. if no new datagram burst blocks arrive on the queue by that time. This ensures detection of stream-sync state changes by the res-sync process even in cases where there is no datagram traffic on the satellite channel. Before the process initiates any processing pass, it tries to obtain a datagram burst block that has a sequence number that is "later" than that of the latest previously received sequence number. This operation may require repeated polling of the "received datagrams to sync" queue. The first block found on the queue that meets this criterion will be used in further processing; intervening blocks will have been discarded. If no such block is found, all blocks on the queue will have been discarded. Although a processing pass will be made in either case, only in the former case is the local queue of past schedulings searched for an element that matches the received datagram burst block. This can limit the processing requirements on the res-sync process and is acceptable since, in most cases, bursts received later in time contain information that better reflects the site's current res-sync state than do earlier bursts.

### 2.4.2.2 Procedure Process_by_State()

The procedure Process by_State() is called once per processing pass. This process reads the CPM status variable one time for the pass to determine the current state of the site on the channel. State transitions effected by other processes are detected by comparing this status value with the value saved from the previous pass. If the site is currently out of stream-sync, the only processing done for the pass is to discard all reservation block chains found on the "scheduled datagrams to sync" queue and put all elements on the local queue of past schedulings onto a free element queue. Otherwise, if in stream-sync, both the local queue and the array of cumulative non-datagram-subframe times are updated using all of the reservation block chains found on the queue from the scheduler. Processing then dispatches to either procedure Out_of_Sync() or In_Sync() depending on the current res-sync state.

### 2.4.2.3 Procedure Out_of_Sync()

The principal function of procedure Out_of_Sync() is to determine of when transitions into the res-sync state should occur, i.e., the acquisition of res-sync. Res-sync is acquired in one of three ways. If there are elements on the local queue of past schedulings and a datagram burst block has been received that is

later than the last previously processed block, the queue is searched for an element which matches the block's sending site and burst ID. If a match is found, the scheduled and observed channel time of the datagram burst are compared. If the two times are the same, thereby verifying correct scheduling, or if the comparison results in the determination and forwarding of a scheduling error to the scheduler process, res-sync has been acquired. Res-sync is also acquired if an interval of (currently) 380 milliseconds elapses, during which BOTH the local queue remains empty and no later datagram burst blocks are received. When res-sync is acquired, Out_of_Sync() sets the res-sync flag in the CPM status variable, thereby informing the remainder of the CPM of the state change.

### 2.4.2.4 Procedure In_Sync()

The principal function of procedure In_Sync() is the detection of past scheduling errors and the forwarding of such errors to the scheduler process, i.e., the maintenance of res-sync. Like procedure Out_of_Sync(), it searches the local queue for an element matching any "later" datagram burst blocks received and does a scheduled time vs. observed time comparison when matches are found. Any past scheduling errors resulting from the comparison are sent to the scheduler. The secondary function of procedure In_Sync() is to detect the situation where a site in

the res-sync state on a non-empty channel does not verify a single correct scheduling for an interval of (currently) 30 seconds. If such a situation occurs, In_Sync() informs the remainder of the CPM of a loss of res-sync by clearing the res-sync flag in the CPM status variable. A res-sync process internal state flag is also set so that the re-acquisition of res-sync within procedure Out_of_Sync() will be deferred until new datagram reservation blocks are received; blocks having been generated by the scheduler process after it recognized the loss of res-sync.

### 2.4.2.5 Function Search_skedQ()

The function Search_skedQ() is the common utility routine used by procedures Out_of_Sync() and In_Sync() to perform searches of the local queue of scheduling history. The function returns one of three result codes. One code indicates that a matching queue element was found with a scheduled time that was the same as the datagram burst's observed time. Another code indicates that a match was found, but that a local scheduling error was determined and sent to the scheduler. In either of these two cases, all local queue elements, from the head element, through the matched element are removed from the queue and are returned to a free element queue. The third code indicates that the local queue search yielded no conclusive information. This can happen if: a

matching element is not found on the queue, the matching element has a scheduled time which is later than the observed time, indicating that the site that transmitted the observed datagram burst requires a scheduling correction, or there was insufficient scheduler history or other problems encountered in the determination of any scheduling error. In the latter two cases, the element in question is spliced out of the local queue and returned to the free element queue.

#### 2.4.2.6 Error Outputs to Process Scheduler

Function Search_skedQ() uses a structure in the CPM common memory segment to pass scheduling error information to the scheduler process. The structure has two entries: the extent of the error in the aggregate datagram subframe, and the frame number of the frame in which the incorrect scheduling was made. The first entry includes compensation for non-datagram-subframe channel time that occurs between datagram subframes when the scheduled and observed transmission times are found to be in different PODA frames. Such compensation is calculated with the aid of the array of cumulative non-datagram subframe times received from the scheduler. The second entry is used by the scheduler in its determination of the amount of scheduling adjustment, if any, that the corresponding error entry requires. The scheduler makes this determination at its next available datagram scheduling

point. In determining the scheduling adjustment, the scheduler also uses its internally stored values for the cumulative unused datagram-subframe channel time for the current frame and for the frame of the incorrect scheduling. The latter value is fetched from an array of such values maintained for the last 32 (currently) frames scheduled. Because the scheduler process performs the calculation of the actual required scheduling adjustment, there is no need for wait/lock synchronization between the scheduler and the res-sync process.

Integrity of the entries in the CPM common memory segment scheduling error structure is ensured via the use of atomic block transfers when reading or writing the data. The scheduler zeroes the error value after it fetches a non-zero error. Also, the res-sync process simply overwrites old error information, that has not yet been processed by the scheduler with any newly detected error information. The new information more accurately reflects the current state of datagram scheduling on the channel.

## DISTRIBUTION

ARPA
Director (3 copies)
Defense Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209
Attn: Program Manager
R. Kahn
R. Ohlander
B. Leiner

DEFENSE DOCUMENTATION CENTER (12 copies)
Cameron Station
Alexandria, VA 22314

DEFENSE COMMUNICATIONS ENGINEERING CENTER
1860 Wiehle Road
Reston, VA 22090
Attn: Maj. J. Fredricks

DEPARTMENT OF DEFENSE
9800 Savage Road
Ft. Meade, MD 20755
Attn: R. McFarland C132 (2 copies)

DEFENSE COMMUNICATIONS AGENCY
8th and South Courthouse Road
Arlington, VA 22204
Attn: Code B645
Glynn Parker, Code B626

NAVAL ELECTRONIC SYSTEMS COMMAND
Department of the Navy
Washington, DC 20360
Attn: B. Hughes, Code 6111
F. Deckelman, Code 6131

MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02138
Attn: D. Clark

MIT Lincoln Laboratory
244 Woods Street
Lexington, MA 02173
Attn: C. Weinstein

DISTRIBUTION cont'd

<u>USC Information Sciences Institute</u>
4676 Admiralty Way
Marina Del Rey, CA 90291
Attn: D. Cohen
S. Casner

DISTRIBUTION cont'd

BOLT BERANEK AND NEWMAN INC.
1300 North 17th Street
Arlington, VA 22209
Attn: E. Wolf

BOLT BERANEK AND NEWMAN INC.
10 Moulton Street
Cambridge, MA 02238

S. Blumenthal
M. Brescia
R. Bressler
T. Calderwood
P. Cudhea
A. Echenique
R. Edmiston
W. Edmond
G. Falk
J. Goodhue
S. Groff
R. Gurwitz
J. Haverty
F. Heart
J. Herman
R. Hinden
D. Hunt
S. Kent
J. Malman
A. McKenzie
D. Melone
W. Milliken
M. Nodine
R. Rettberg
H. Rising
J. Robinson
E. Rosen
P. Santos
S. Storch
R. Thomas
C. Topolcic
R. Waters
B. Woznick
Library